graphical button **210**, a general warning, e.g., "Improve your level of performance," is sent to Player 1. In another embodiment, when the spectator clicks on graphical button **210**, an interface is displayed that enables the spectator either to select a general warning or to generate a more specific warning, e.g., "Start passing more and stop taking so many jump shots." In one embodiment, the specific warning is generated by inserting text in a graphical control element, e.g., a text box.

Graphical button **212**, which is labeled "Provide Custom Message," enables the spectator to send a custom message to Player 1. By way of example, the custom message can be a compliment for the player, e.g., "Nice game today" or "You shoot the ball really well," or constructive criticism for the player, e.g., "You need to work on your jump shot." In one embodiment, when the spectator clicks on graphical button **212**, an interface is displayed that enables the spectator to generate the custom message by inserting text in a graphical control element, e.g., a text box.

Graphical button **214**, which is labeled "Keep in Game," enables the spectator to cast a vote to keep Player 1 in the game. By way of example, the spectator might want to vote for Player 1 to stay in the game because the spectator likes the way Player 1 plays the game. Alternatively, the spectator might want to vote to keep Player 1 in the game to prevent other spectators from getting enough votes to have Player 1 removed from the game, as will explained in more detail below.

The input received through voting interface **206** is transmitted to be processed into a crowd sourced input message **216**. In the case of votes either to keep a player in the game or to remove a player from the game, in one embodiment, each vote is weighted in accordance with the skill level of the person casting the vote. For example, when spectator S_1 casts a vote, spectator S_1's skill level **218** is included in the processing used to generate crowd sourced input message **216**. A person's skill level **218** reflects the history of the person with respect to the particular game being played. By way of example, if spectator S_1 is watching a basketball game and casts a vote regarding a player, spectator S_1's vote would be weighted based on the skill reflected by metrics in spectator S_1's game profile for the basketball game. By way of example, the metrics in a spectator's game profile that can be included in assessing the level of skill can include how often a person plays the game, the person's game statistics (e.g., points, goals, assists, etc.), the person's game rating, and the accomplishments of the person in the game (e.g., levels achieved, trophies won, etc.).

In one embodiment, votes by spectators having relatively high skill levels in a game are given more weight than votes by spectators having relatively low skill levels in the game. This makes it more difficult for spectators having relatively low skill levels in a game to unduly influence whether players are either kept in or removed from the game. In one embodiment, votes by spectators having a relatively low skill level are counted once (no extra weight given), votes by spectators having an average level of skill are counted several times (e.g., 2× or 3×), and votes by spectators having a relatively high level of skill are counted many times (e.g., 5× or 10× or even higher). Thus, if spectator S_1 is watching a basketball game and spectator S_1's skill level **218** indicates that spectator S_1 has a relatively high skill level in the basketball game, spectator S_1's vote to remove a player from the basketball game would be given, in one embodiment, ten times more weight than a vote from a spectator having a relatively low skill level in the basketball game.

As shown in FIG. **2**, spectator S_4 is also watching the live game via user interface **200** displayed on spectator S_4's client device, e.g., a computer, tablet, smartphone, etc. As discussed above with reference to spectator S_1, the user interface **200** includes the game view **202**, the communication channel **202**, and the voting interface **206**. As shown in FIG. **2**, spectator S_4 has clicked on the graphical button labeled "Player 3" in the voting interface **206** to cause a player voting interface **206-3** for Player 3 to be displayed. The player voting interface **206-3** for Player 3 includes the same graphical buttons described above with reference to player voting interface **206-1** for Player 1, namely, graphical buttons **208** ("Remove Player from Game"), **210** ("Warn Player to Improve"), **212** ("Provide Custom Message"), and **214** ("Keep in Game"). Although not shown in FIG. **2**, when a spectator clicks on the graphical button in voting interface **206** for one of the other players in the game, e.g., Player 2, Player 4, . . . Player N, a similar player voting interface is displayed for that player.

As described above with reference to spectator S_1, when spectator S_4 casts a vote, spectator S_4's skill level **218** is included in the processing used to generate crowd sourced input message **216**. Thus, if spectator S_4 is watching a basketball game and spectator S_4's skill level **218** indicates that spectator S_4 has an average skill level in the basketball game, spectator S_4's vote to remove a player from the basketball game would be given, in one embodiment, three times more weight than a vote from a spectator having a relatively low skill level in the basketball game.

The crowd sourced input messages **216** are transmitted to input aggregator **218** for further processing. As shown in FIG. **2**, the crowd sourced input messages **216** from spectator S_1 and spectator S_4 are transmitted to input aggregator **218**. The input aggregator **218** collects the crowd sourced input messages **216** received from the spectators and processes them as needed. In the case of votes either to remove a player from a game or to keep a player in the game, the input aggregator **218** transmits these votes to crowd sourced vote tabulator **220** for further processing, as will be explained in more detail below. In the case of messages to be sent to players in the game, e.g., a general warning or a custom message, the input aggregator **218** performs the functionality required to forward each message to the player for which the message is intended.

The crowd sourced vote tabulator **220** tallies the votes received from the input aggregator **218** and displays the voting results for each player via a suitable user interface **220-1**. As shown in FIG. **2**, the voting results for Player 1 are "Keep in Game" 76%, "Remove from Game" 24%. The voting results for Player 2 are "Keep in Game" 52%, "Remove from Game" 48%. The voting results for Player 3 are "Keep in Game" 34%, "Remove from Game" 66%. Once the crowd sourced vote tabulator **220** has tallied the votes and displayed the voting results, the crowd sourced vote tabulator **220** transmits the voting results to rules engine **222** for further processing to determine whether any players should be removed from the game based on the voting results.

The rules engine **222** applies a set of rules to the voting results for each player to determine whether the player should be removed from the game. By way of example, the set of rules can include rules regarding the percentage of votes required to automatically remove a player from a game, the minimum number of spectators that must vote before a player can be removed from a game, etc. In one embodiment, the threshold percentage of votes required to automatically remove a player from a game is 60%. Thus,